F

Unicode®

F.1 Introdução

O uso de **codificações de caracteres** inconsistentes (valores numéricos associados a caracteres) no desenvolvimento de produtos de softwares globais causa problemas sérios porque os computadores processam informações utilizando números. Por exemplo, o caractere 'a' é convertido em um valor numérico de modo que um computador possa manipular esse fragmento de dados. Muitos países e empresas desenvolveram sistemas de codificação incompatíveis com os de outros países e empresas. Por exemplo, o sistema operacional Microsoft Windows atribui o valor `0xC0` ao caractere 'A com um acento grave', enquanto o sistema operacional Apple Macintosh atribui o mesmo valor a um ponto de interrogação de ponta-cabeça. Isso resulta em representação errônea e possível corrupção de dados.

Na ausência de um padrão universal de codificação de caracteres, desenvolvedores de softwares globais têm de fazer um extenso processo de localização dos seus produtos antes da distribuição. A **localização** inclui tradução entre idiomas e adaptação cultural do conteúdo. O processo de localização normalmente abrange modificações significativas no código-fonte (por exemplo, conversão de valores numéricos e suposições subjacentes feitas pelos programadores), o que resulta em aumento de custos e atrasos na distribuição do software. Por exemplo, um programador que fale inglês projetaria um produto de software global supondo que um caractere único poderia ser representado por um byte. Entretanto, quando a localização desses produtos é feita nos mercados asiáticos, as suposições do programador não mais são válidas porque há muitos mais caracteres asiáticos e, portanto, a grande parte do código, se não todo ele, precisaria ser reescrita. A localização é necessária em cada distribuição de versão. No momento em que ocorre a localização de um produto de software para um mercado em particular, uma versão mais recente, que também precisa ser localizada, pode estar pronta para distribuição. Como resultado, é trabalhoso e caro produzir e distribuir produtos de softwares globais em um mercado em que não haja nenhum padrão universal de codificação de caracteres.

Em resposta a essa situação, foi criado o **Padrão Unicode**, um padrão de codificação que facilita a produção e distribuição de softwares. O Padrão Unicode delineia uma especificação para produzir uma codificação consistente dos caracteres e símbolos mundiais. Deve ocorrer a localização de produtos de softwares que tratam texto codificado no Padrão Unicode, mas o processo de localização é mais simples e mais eficiente porque os valores numéricos não precisam ser convertidos e as suposições feitas pelos programadores sobre a codificação de caracteres são universais. O Padrão Unicode é mantido por uma organização sem fins lucrativos chamada **Unicode Consortium**, cujos membros incluem a Apple, IBM, Microsoft, Oracle, Sun Microsystems, Sybase e muitos outros.

Quando o Consortium pensou e desenvolveu o Padrão Unicode, ele queria um sistema de codificação que fosse **universal**, **eficiente**, **uniforme** e **não ambíguo**. Um sistema de codificação universal inclui todos os caracteres comumente utilizados. Um sistema de codificação eficiente permite que arquivos de texto sejam analisados sintaticamente com rapidez. Um sistema de codificação uniforme atribui valores fixos a todos os caracteres. Um sistema de codificação não ambíguo representa um dado caractere de uma maneira consistente. Esses quatro termos são referidos como a base de design do Padrão Unicode.

F.2 Formatos de transformação Unicode

Embora o Unicode incorpore o conjunto de caracteres ASCII limitado (ou seja, uma coleção de caracteres), ele inclui um conjunto de caracteres mais abrangente. Em ASCII cada caractere é representado por um byte contendo 0s e 1s. Um byte é capaz de armazenar os números binários de 0 a 255. A cada caractere é atribuído um número entre 0 e 255, portanto sistemas baseados em ASCII suportam somente 256 caracteres, uma pequena fração dos caracteres mundiais. O Unicode estende o conjunto de caracteres ASCII codificando a ampla maioria dos caracteres mundiais. O Padrão Unicode codifica caracteres em um espaço numérico uniforme entre 0 a 10FFFF hexadecimal. Uma implementação expressará esses números em um dos vários formatos de transformação, selecionando o que melhor se ajusta ao aplicativo particular disponível.

Esses três formatos em uso são chamados **UTF-8**, **UTF-16** e **UTF-32**. O UTF-8, uma forma de codificação de largura variável, requer de um a quatro bytes para expressar cada caractere Unicode. Dados em UTF-8 consistem em bytes de 8 bits (seqüências de um,

dois, três ou quatro bytes dependendo do caractere em codificação) e são bem adequados para sistemas baseados em ASCII quando há uma predominância de caracteres de um byte (o ASCII representa caracteres como de um byte). Hoje, o UTF-8 é amplamente implementado em sistemas UNIX e em bancos de dados.

A forma de codificação UTF-16 de largura variável expressa os caracteres Unicode em unidades de 16 bits (como dois bytes adjacentes ou um inteiro curto em muitas máquinas). A maioria dos caracteres Unicode é expressa em uma única unidade de 16 bits. Entretanto, caracteres com valores acima do hexadecimal FFFF são expressos com um par ordenado de unidades de 16 bits chamado **substitutos**. Os substitutos são inteiros de 16 bits no intervalo ente D800 a DFFF, unicamente utilizados para o propósito de 'proteger' (escape) os caracteres numerados mais altos. Cerca de um milhão de caracteres podem ser expressos dessa maneira. Embora um par de substitutos exija 32 bits para representar caracteres, ele apresenta espaço eficiente para utilizar essas unidades de 16 bits. Os substitutos são caracteres raros nas implementações atuais. Muitas implementações de tratamento de strings são escritas em termos do UTF-16. [*Nota:* Os detalhes e o código de exemplo para tratamento UTF-16 estão disponíveis no site da Web do Unicode Consortium em `www.unicode.org`].

As implementações que requerem uso significativo de caracteres raros ou scripts inteiros codificados acima do hexadecimal FFFF devem utilizar o UTF-32, uma forma de codificação de largura fixa de 32 bits que normalmente exige duas vezes mais memória do que os caracteres codificados em UTF-16. A principal vantagem da forma de codificação UTF-32 de largura fixa é que ela expressa todos os caracteres uniformemente, portanto é fácil de tratá-la em arrays.

Há poucas diretrizes que afirmam quando utilizar determinada codificação. A melhor forma de codificação a utilizar depende do sistema de computador e do protocolo de negócios, não dos próprios dados. Em geral, a codificação UTF-8 deve ser utilizada em sistemas de computador e protocolos de negócios que requerem que os dados sejam tratados em unidades de 8 bits, particularmente em sistemas legados em utilização, pois isso freqüentemente simplifica modificações nos programas existentes. Por essa razão, o UTF-8 tornou-se a codificação preferida na Internet. Da mesma maneira, o UTF-16 é a codificação preferida nos aplicativos Microsoft Windows. Há probabilidade de o UTF-32 vir a ser mais amplamente utilizado no futuro à medida que mais caracteres são codificados com valores acima do hexadecimal FFFF. O UTF-32 requer tratamento menos sofisticado que o UTF-16 na presença de pares substitutos. A Figura F.1 mostra as diferentes maneiras pelas quais os três tipos de codificação tratam a codificação de caracteres.

F.3 Caracteres e glifos

O Padrão Unicode consiste em caracteres — componentes escritos (alfabetos, números, marcas de pontuação, marcas de acento etc.) — que podem ser representados por valores numéricos. Um exemplo desse caractere é a LETRA MAIÚSCULA LATINA A, U+0041. Na primeira representação de caractere, **U+*aaaa*** é um **valor de código**, em que U + se refere a valores de código Unicode, em oposição a outros valores hexadecimais. O *aaaa* representa um número de quatro dígitos hexadecimais de um caractere codificado. Os valores de código são combinações de bits que representam caracteres codificados. Os caracteres são representados utilizando-se **glifos** — várias formas, fontes e tamanhos para exibir caracteres. Não há nenhum valor de código para glifos no Padrão Unicode. Os exemplos de glifos são mostrados na Figura F.2.

Caractere	UTF-8	UTF-16	UTF-32
LETRA MAIÚSCULA LATINA A	0x41	0x0041	0x00000041
LETRA GREGA MAIÚSCULA ALFA	0xCD 0x91	0x0391	0x00000391
IDEOGRAMA-4E95 UNIFICADO de CJK	0xE4 0xBA 0x95	0x4E95	0x00004E95
LETRA ITÁLICA ANTIGA A	0xF0 0x80 0x83 0x80	0xDC00 0xDF00	0x00010300

Figura F.1 Correlação entre as três formas de codificação.

Figura F.2 Vários glifos do caractere A.

O Padrão Unicode inclui os alfabetos, ideogramas, lista de silabas, marcas de pontuação, **diacríticos**, operadores matemáticos e outros recursos que abrangem os idiomas escritos e manuscritos mundiais. O diacrítico é uma marca especial adicionada ao caractere para distingui-lo de uma outra letra ou para indicar um acento (por exemplo, em espanhol, o til '~' acima do caractere 'n'). Atualmente, o Unicode fornece valores de código para 96.382 representações de caracteres, com mais de 878 mil valores de código reservados para expansão futura.

F.4 Vantagens e desvantagens do Unicode

O Padrão Unicode tem várias vantagens significativas que promovem seu uso. Uma delas é o impacto que tem sobre o desempenho da economia internacional. O Unicode padroniza os caracteres para os sistemas mundiais de escrita de acordo com um modelo uniforme que promove a transferência e compartilhamento de dados. Os programas desenvolvidos com esse esquema mantêm sua exatidão porque cada

caractere tem uma única definição (*a* é sempre U+0061, % é sempre U+0025). Isso permite que as empresas administrem as altas demandas dos mercados internacionais processando diferentes sistemas de escrita ao mesmo tempo. Todos os caracteres podem ser gerenciados de maneira idêntica, evitando assim qualquer confusão causada por diferentes arquiteturas de códigos de caracteres. Além disso, gerenciar dados de maneira consistente elimina a corrupção de dados, pois eles podem ser classificados, pesquisados e manipulados utilizando-se um processo consistente.

Uma outra vantagem do Padrão Unicode é a portabilidade softwares que podem executar em diferentes computadores ou em diferentes sistemas operacionais). A maioria dos sistemas operacionais, bancos de dados, linguagens de programação (incluindo o Java e linguagens .NET da Microsoft) e navegadores da Web atualmente suportam, ou estão planejando suportar, o Unicode.

Uma desvantagem do Padrão Unicode é a quantidade de memória requerida pelo UTF-16 e UTF-32. Os conjuntos de caracteres ASCII têm comprimento de 8 bits, portanto requerem menos espaço de armazenamento do que os conjuntos de caracteres Unicode, de 16 bits padrão. O **conjunto de caracteres de dois bytes** (**double-byte character set – DBCS**) codifica caracteres asiáticos com um ou dois bytes por caractere. O **conjunto de caracteres de multibyte** (**multibyte character set – MBCS**) codifica caracteres com um número variável de bytes por caractere. Nessas instâncias, as formas de codificação UTF-16 ou UTF-32 podem ser utilizadas com pouco impacto sobre a memória e desempenho.

Uma outra desvantagem do Unicode é que, embora inclua mais caracteres do que qualquer outro conjunto de caracteres em uso comum, ele ainda não codifica todos os caracteres escritos mundiais. Além disso, o UTF-8 e UTF-16 são tipos de codificação de largura variável, assim os caracteres ocupam diferentes quantidades de memória.

F.5 Site da Web do Unicode Consortium

Se quiser aprender mais sobre o Padrão Unicode, visite `www.unicode.org`. Esse site fornece várias informações úteis sobre o Padrão Unicode para iniciantes ao Unicode. A home page é organizada em **Announcements**, **New to Unicode**, **General Information**, **The Consortium**, **For Members Only**, **The Unicode Standard**, **Key Specifications**, **Technical Publications** e **Work in Progress**.

A seção Announcement lista as atualizações dos produtos recentes, distribuições e revisões públicas; ela lista ainda novos membros do Unicode Consortium.

A seção **New to Unicode** consiste em quatro subseções: **What is Unicode**, **How to Use this Site**, **FAQ** e **Glossary of Unicode Terms**. A primeira subseção fornece uma introdução técnica ao Unicode descrevendo os princípios de projeto, interpretações e atribuições de caracteres, processamento de textos e conformidade com o Unicode. A leitura dessa subseção é recomendada para qualquer iniciante ao Unicode. Ela também inclui uma lista de links relacionados que pode fornecer ao leitor informações adicionais sobre o Unicode. A subseção **How to Use this Site** contém informações sobre a utilização e navegação pelo site e também hyperlinks para recursos adicionais. A subseção **FAQ** organiza perguntas feitas com freqüência em vários tópicos. Cada tópico contém uma explicação breve que especifica quais os tipos de perguntas e respostas são fornecidos. Leitores pouco familiarizados com os termos utilizados pelo Unicode Consortium podem navegar pela subseção **Glossary of Unicode Terms**, que lista os termos Unicode e suas definições em ordem alfabética.

A seção **General Information** contém seis subseções: **Where is my Character**, **Display Problems**, **Useful Resources**, **Unicode Enabled Products**, **Mail Lists** e **Conferences**. As principais áreas abrangidas nessa seção incluem um link aos gráficos de códigos Unicode (uma listagem completa dos valores de código) montados pelo Unicode Consortium, bem como uma estrutura de tópicos detalhada sobe como localizar um caractere codificado no gráfico de códigos. A seção também contém recomendações sobre como configurar diferentes sistemas operacionais e navegadores da Web de modo que os caracteres Unicode possam ser visualizados adequadamente. Além disso, nessa seção, o usuário pode navegar para outros sites que fornecem informações sobre tópicos como, fontes, lingüística e outros padrões, como a **Armenian Standards Page** e o **Chinese GB 18030 Encoding Standard**.

A seção **The Consortium** consiste em seis subseções: **Who we are**, **Our Members**, **How to Join**, **Press Info**, **Policies & Positions** e **Contact Us**. Essa seção fornece uma lista dos membros atuais do Unicode Consortium, bem como informações sobre como tornar-se um membro. Os privilégios para cada tipo de membro — integral, associado, especialista, individual e de ligação — e as taxas para cada membro são listados aqui.

A seção **For Members Only** consiste em duas subseções: **Member Resources** e **Working Documents**. Essas subseções são protegidas por senha — somente os membros do consórcio podem acessar esses links.

A seção The **Unicode Standard** consiste em cinco subseções: **Start Here**, **Latest Version**, **Code Charts**, **Unicode Character Database** e **Unihan Database**. Essa seção descreve as atualizações aplicadas à versão mais recente do Padrão Unicode bem como para categorizar todas as codificações definidas. O usuário pode aprender como a versão mais recente foi modificada a fim de incluir mais recursos e capacidades. Por exemplo, um aprimoramento da Versão 4.0 é o fato de ela conter caracteres codificados adicionais.

A seção **Key Specification** consiste em cinco subseções: **Unicode Collation (UCA)**, **Bidirectional Algorithm (Bidi)**, **Normalization (NFC, NFD, L...A l)**, **Locale Data (CLDR)** e **Scripts Codes (ISO 15924)**. Elas descrevem as especificações e projetos chave relacionados ao Unicode.

A seção **Technical Publication** consiste em quatro subseções: **Technical Reports & Standards**, **Technical Notes**, **Online Data Table** e **Updates & Errata**. A subseção Technical Reports & Standards contém relatórios e padrões utilizados para implementar e desenvolver o padrão Unicode. A subseção **Technical Notes** lista artigos nos quais os usuários ou implementadores do Unicode podem estar interessados. A subseção **Online Data Table** fornece tabelas legíveis por máquina que são requeridas para implementar o padrão Unicode. A subseção **Updates & Errata** lista erratas conhecidas para a versão atual, que serão corrigidas na próxima versão.

A seção **Work in Progress** consiste em seis subseções: **Calendar of Meetings**, **Unicode Technical Committee**, **Meeting Minutes**, **Proposals for Public Review**, **Proposed Characters** e **Submitting Proposals**. Ela apresenta ao usuário os caracteres no catálogo

recentemente incluídos ao esquema do Padrão Unicode, bem como caracteres considerados para inclusão. O usuário que determinar que um caractere foi menosprezado pode submeter uma proposta por escrito para a inclusão desse caractere. A subseção **Submitting Proposals** contém diretrizes estritas que devem ser obedecidas ao submeter propostas por escrito.

F.6 Utilizando o Unicode

Várias linguagens de programação (por exemplo, C, Java, JavaScript, Perl, Visual Basic) fornecem algum nível de suporte para o Padrão Unicode. O aplicativo mostrado nas figuras F.3 e F.4 imprime o texto "Bem-vindo ao Unicode!" em oito diferentes idiomas: inglês, russo, francês, alemão, japonês, português, espanhol e chinês tradicional.

```
1  // Fig. F.3: UnicodeJFrame.java
2  // Demonstrando como utilizar o Unicode em programas Java.
3  import java.awt.GridLayout;
4  import javax.swing.JFrame;
5  import javax.swing.JLabel;
6
7  public class UnicodeJFrame extends JFrame
8  {
9     // construtor cria JLabels para exibir Unicode
10    public UnicodeJFrame()
11    {
12       super( "Demonstrating Unicode" );
13
14       setLayout( new GridLayout( 8, 1 ) ); // configura o layout de frame
15
16       // cria JLabels utilizando o Unicode
17       JLabel englishJLabel = new JLabel( "\u0057\u0065\u006C\u0063" +
18          "\u006F\u006D\u0065\u0020\u0074\u006F\u0020Unicode\u0021" );
19       englishJLabel.setToolTipText( "This is English" );
20       add( englishJLabel );
21
22       JLabel chineseJLabel = new JLabel( "\u6B22\u8FCE\u4F7F\u7528" +
23          "\u0020\u0020Unicode\u0021" );
24       chineseJLabel.setToolTipText( "This is Traditional Chinese" );
25       add( chineseJLabel );
26
27       JLabel cyrillicJLabel = new JLabel( "\u0414\u043E\u0431\u0440" +
28          "\u043E\u0020\u043F\u043E\u0436\u0430\u043B\u043E\u0432" +
29          "\u0430\u0442\u044A\u0020\u0432\u0020Unicode\u0021" );
30       cyrillicJLabel.setToolTipText( "This is Russian" );
31       add( cyrillicJLabel );
32
33       JLabel frenchJLabel = new JLabel( "\u0042\u0069\u0065\u006E\u0076" +
34          "\u0065\u006E\u0075\u0065\u0020\u0061\u0075\u0020Unicode\u0021");
35       frenchJLabel.setToolTipText( "This is French" );
36       add( frenchJLabel );
37
38       JLabel germanJLabel = new JLabel( "\u0057\u0069\u006C\u006B\u006F" +
39          "\u006D\u006D\u0065\u006E\u0020\u007A\u0075\u0020Unicode\u0021");
40       germanJLabel.setToolTipText( "This is German" );
41       add( germanJLabel );
42
43       JLabel japaneseJLabel = new JLabel( "Unicode\u3078\u3087\u3045" +
44          "\u3053\u305D\u0021" );
45       japaneseJLabel.setToolTipText( "This is Japanese" );
46       add( japaneseJLabel );
```

Figura F.3 Aplicativo Java que utiliza a codificação Unicode. (Parte 1 de 2.)

```
47
48       JLabel portugueseJLabel = new JLabel( "\u0053\u00E9\u006A\u0061" +
49          "\u0020\u0042\u0065\u006D\u0076\u0069\u006E\u0064\u006F\u0020" +
50          "Unicode\u0021" );
51       portugueseJLabel.setToolTipText( "This is Portuguese" );
52       add( portugueseJLabel );
53
54       JLabel spanishJLabel = new JLabel( "\u0042\u0069\u0065\u006E" +
55          "\u0076\u0065\u006E\u0069\u0064\u0061\u0020\u0061\u0020" +
56          "Unicode\u0021" );
57       spanishJLabel.setToolTipText( "This is Spanish" );
58       add( spanishJLabel );
59    } // fim do construtor UnicodeJFrame
60 } // fim da classe UnicodeJFrame
```

Figura F.3 Aplicativo Java que utiliza a codificação Unicode. (Parte 2 de 2.)

```
1  // Fig. F.4: Unicode.Java
2  // Exibe Unicode.
3  import javax.swing.JFrame;
4
5  public class Unicode
6  {
7     public static void main( String args[] )
8     {
9        UnicodeJFrame unicodeJFrame = new UnicodeJFrame();
10       unicodeJFrame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
11       unicodeJFrame.setSize( 350, 250 );
12       unicodeJFrame.setVisible( true );
13    } // fim do método main
14 } // fim da classe Unicode
```

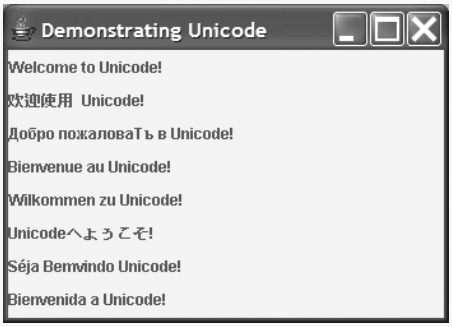

Figura F.4 Exibindo Unicode.

A classe `UnicodeJFrame` (Figura F.3) usa seqüências de escape para representar os caracteres. Uma seqüência de escape na forma `\u`*yyyy*, onde *yyyy* representa o valor do código de hexadecimal de quatro dígitos. As linhas 17—18 contêm a série de seqüências de escape necessária para exibir "Welcome to Unicode!" em inglês. A primeira seqüência de escape (`\u0057`) é igualada ao caractere 'W', a segunda seqüência de escape (`\u0065`) é igualada ao caractere 'e' e assim por diante. A seqüência de escape `\u0020` (linha 18) é a codificação para o caractere de espaço em branco. As seqüências de escape `\u0074` e `\u006F` são igualadas à palavra 'to'. Observe que 'Unicode' não é codificado porque é uma marca comercial registrada e não tem nenhuma tradução equivalente na maioria dos idiomas. A linha 18 também contém a seqüência de escape `\u0021` para o ponto de exclamação (`!`).

As linhas 22—56 contêm as seqüências de escape para os outros sete idiomas. O site da Web do Unicode Consortium contém um link para gráficos de códigos que lista os valores de códigos Unicode de 16 bits. Caracteres ingleses, franceses, alemães, portugueses e espanhóis estão localizados no bloco **Basic Latin**, os caracteres japoneses estão localizados no bloco **Hiragana**, os caracteres russos estão localizados no bloco **Cyrillic** e os caracteres chineses tradicionais estão localizados no bloco **CJK Unified Ideographs**. A seção a seguir discute esses blocos.

F.7 Intervalos de caracteres

O Padrão Unicode atribui valores de código, que variam de `0000` (**Basic Latin**) a `E007F` (**Tags**), para os caracteres escritos mundiais. Atualmente, há valores de códigos para 96.382 caracteres. Para simplificar a pesquisa de um caractere e seu valor de código associado, em geral o Padrão Unicode agrupa valores de código por **script** e função (isto é, caracteres latinos são agrupados em um bloco, operadores matemáticos são agrupados em um outro bloco etc.). De modo geral, um script é um único sistema de escrita utilizado por múltiplos idiomas (por exemplo, o script Latin é utilizado pelo inglês, francês, espanhol etc.). A página **Code Charts** no site da Web do Unicode Consortium lista todos os blocos definidos e seus respectivos valores de código. A Figura F.5 lista alguns blocos (scripts) no site da Web e seus intervalos dos valores de código.

Script	Intervalo dos valores de código
Árabe	U+0600–U+06FF
Latino básico	U+0000–U+007F
Bengali (Índia)	U+0980–U+09FF
Cherokee (América indígena)	U+13A0–U+13FF
Ideogramas unificados de China, Japão e Coréia (Ásia oriental)	U+4E00–U+9FFF
Cirílico (Rússia e Europa Oriental)	U+0400–U+04FF
Etíope	U+1200–U+137F
Grego	U+0370–U+03FF
Hangul Jamo (Coréia)	U+1100–U+11FF
Hebraico	U+0590–U+05FF
Hiragana (Japão)	U+3040–U+309F
Khmer (Camboja)	U+1780–U+17FF
Lao (Laos)	U+0E80–U+0EFF
Mongol	U+1800–U+18AF
Birmanês	U+1000–U+109F
Ogham (Irlanda)	U+1680–U+169F
Rúnico (Alemanha e Escandinávia)	U+16A0–U+16FF
Sinhala (Sri Lanka)	U+0D80–U+0DFF
Telugu (Índia)	U+0C00–U+0C7F
Tai	U+0E00–U+0E7F

Figura F.5 Alguns intervalos de caracteres.

Resumo

- Antes do Unicode, desenvolvedores de software foram atormentados pelo uso de codificação de caracteres inconsistente (isto é, valores numéricos para caracteres). A maioria dos países e empresas tinha seus próprios sistemas de codificação, que eram incompatíveis.
- A localização de softwares globais exige modificações significativas no código-fonte, resultando no aumento de custos e atrasos na distribuição do produto.
- A localização é necessária em cada distribuição de uma versão. No momento em que ocorre a localização de um produto de software para um mercado em particular, uma versão mais recente, que também precisa ser localizada, está pronta para distribuição Como resultado, é trabalhoso e caro produzir e distribuir produtos de software globais em um mercado em que não há nenhum padrão universal de codificação de caracteres.
- O Unicode Consortium desenvolveu o Padrão Unicode em resposta aos sérios problemas criados por múltiplas codificações de caracteres e uso dessas codificações.
- O Padrão Unicode facilita a produção e distribuição de software localizado. Ele descreve uma especificação para codificação consistente dos caracteres e símbolos mundiais.
- Produtos de software que tratam texto codificado no Padrão Unicode precisam ser localizados, mas o processo de localização é mais simples e mais eficiente porque não há necessidade de converter os valores numéricos.
- O Padrão Unicode é projetado para ser universal, eficiente, uniforme e não ambíguo.
- Um sistema universal de codificação inclui todos os caracteres comumente utilizados; um sistema eficiente de codificação analisa sintaticamente arquivos de texto de maneira fácil; um sistema uniforme de codificação atribui valores fixos a todos os caracteres; e um sistema não ambíguo de codificação representa o mesmo caractere para qualquer valor dado.
- O Unicode estende o limitado conjunto de caracteres ASCII para incluir todos os caracteres mundiais importantes.
- O unicode utiliza três formatos de transformação Unicode (UTF – *Unicode transformation format*): UTF-8, UTF-16 e UTF-32, cada um dos quais pode ser apropriado para uso em diferentes contextos.
- Dados em UTF-8 consistem em bytes de 8 bits (seqüências de um, dois, três ou quatro bytes dependendo do caractere em codificação) e são bem adequados para sistemas baseados em ASCII quando há uma predominância de caracteres de um byte (o ASCII representa caracteres como de um byte).

- O UTF-8 é uma forma de codificação de largura variável mais compacta para texto que envolve principalmente caracteres latinos e pontuação ASCII.
- O UTF-16 é a forma padrão de codificação do Padrão Unicode. Ele é uma forma de codificação de largura variável que utiliza unidades de 16 bits de código em vez de bytes. A maioria dos caracteres é representada por uma única unidade de 16 bits, mas alguns caracteres requerem pares de substitutos.
- O UTF-32 é uma forma de codificação de 32 bits. A principal vantagem da forma de codificação de largura fixa é que ele expressa todos os caracteres uniformemente, assim são fáceis de tratar em arrays e em outros usos.
- Os caracteres são representados por glifos — formas, fontes e tamanhos para exibir caracteres.
- Os valores de código são combinações de bits que representam caracteres codificados. A notação Unicode para um valor de código é U+*yyyy* na qual U+ se refere a valores de código Unicode, em oposição a outros valores hexadecimais. O *yyyy* representa um número de quatro dígitos hexadecimais.
- Atualmente, o Padrão Unicode fornece valores de código para 96.382 representações de caracteres.
- Uma vantagem do Padrão Unicode é seu impacto sobre o desempenho geral da economia internacional. Os aplicativos em conformidade com um padrão de codificação podem ser processados facilmente pelos computadores.
- Uma outra vantagem do Padrão Unicode é sua portabilidade. Os aplicativos escritos em Unicode podem ser facilmente transferidos para diferentes sistemas operacionais, bancos de dados e navegadores da Web. A maioria das empresas atualmente suporta ou planeja suportar o Unicode.
- Várias linguagens de programação fornecem algum nível de suporte ao Padrão Unicode.
- Nos programas Java, a seqüência de escape \u*yyyy* representa um caractere, onde *yyyy* é o valor de código de quatro dígitos hexadecimais. A seqüência de escape `\u0020` é a codificação universal para o caractere de espaço em branco.

Terminologia

Base de projeto Unicode
bloco
codifica
conjunto de caracteres de dois bytes (DBCS – *double-byte character sets*)
conjunto de caracteres de multibyte (MBCS –*multibyte character set*)
diacrítico
eficiente (base de projeto Unicode)
glifo
localização
não ambíguo (base de projeto Unicode)
notação hexadecimal
Padrão Unicode
portabilidade
script
seqüência de escape \u*yyyy*
substituto
Unicode Consortium
Unicode Transformation Format (UTF)
uniforme (base de projeto Unicode)
universal (base de projeto Unicode)
UTF-16
UTF-32
UTF-8
valor de código

Exercícios de revisão

F.1 Preencha as seguintes lacunas.

a) Desenvolvedores de softwares globais tinham uma _____ de seus produtos para um mercado específico antes da distribuição.
b) O Padrão Unicode é um padrão de _____que facilita a produção e distribuição uniforme de produtos de software.
c) As quatro bases de projetos que abrangem o Padrão Unicode são: _____, _____, _____ e _____.
d) Os caracteres são representados utilizando _____.
e) Diz-se que um software que pode executar em diferentes sistemas operacionais é _____.

F.2 Determine se cada um dos itens a seguir é *verdadeiro* ou *falso*. Se *falso*, explique por quê.

a) O Padrão Unicode inclui todos os caracteres mundiais.
b) Um valor de código Unicode é representado como U+*yyyy*, onde *yyyy* representa um número na notação binária.
c) Um diacrítico é caractere com uma marca especial que enfatiza um acento.
d) O unicode é portável.
e) Ao projetar programas Java, uma seqüência de escape Unicode é denotada por /u*yyyy*.

Respostas dos exercícios de revisão

F.1 a) localização. b) codificação. c) universal, eficiente, uniforme, não ambíguo. d) glifos. e) portável.

F.2 a) Falso. Inclui a maioria dos caracteres mundiais. b) Falso. O *yyyy* representa um número hexadecimal. c) Falso. Um diacrítico é uma marca especial adicionada ao caractere para distingui-lo de outra letra ou para indicar um acento. d) Verdadeiro. e) Falso. Uma seqüência de escape Unicode é representada por \u*yyyy*.

Exercícios

F.3 Navegue pelo site da Web do Unicode Consortium (`www.unicode.org`) e escreva os valores de código em hexadecimal para os caracteres a seguir. Em qual bloco eles estavam localizados?

a) Letra latina 'Z'.
b) Letra latina 'n' com o 'til (~)'.

c) Letra grega ‘delta’.
d) Operador matemático ‘menor que ou igual a’.
e) Símbolo de pontuação ‘aspas abertas (“)’.

F.4 Descreva a base de projeto do Padrão Unicode.

F.5 Defina os termos a seguir:
a) Valor de código.
b) Substitutos.
c) Padrão Unicode.

F.6 Defina os termos a seguir:
a) UTF-8.
b) UTF-16.
c) UTF-32.

F.7 Descreva um cenário em que é ótimo armazenar seus dados no formato UTF-16.

F.8 Utilizando os valores de código do Padrão Unicode, escreva um programa Java que imprima seu primeiro e último nome. O programa deve imprimir seu nome com todas as letras maiúsculas e minúsculas. Se conhecer outros idiomas, imprima seu primeiro e último nome também nesses idiomas.